VIEW: A DISTRIBUTED SYSTEM FOR GRAPHICAL ANALYSIS OF LARGE DATA BASES

Eric F. Harslem, et al

RAND Corporation
Santa Monica, California

March 1973

VIEW: A DISTRIBUTED SYSTEM FOR GRAPHICAL ANALYSIS OF LARGE DATA BASES

Eric F. Harslem
Suzanne D. Landa

March 1973

P-4972

CONTENTS

ABSTRACT

Research projects using large numerical simulations of natural phenomena encounter problems acquiring computer and data storage resources. Work by the ARPA-sponsored climate project at Rand is cited as exemplifying these problems. Certain of these problems are solvable by the large-scale resource sharing of the ARPA Network.

The VIEW system addresses a further problem -- aiding researchers via graphical analysis of large, remotely-located data bases. In order to access remote data storage facilities (e.g., the trillion-bit Laser Store), modules of VIEW are distributed over the ARPA Network, with the main analysis module on the UCLA 360/91. The user/system interface was designed to satisfy a set of user-generated specifications and to allow syntactically different inputs to remote data retrieval systems. Terminal input/output is in a Network standard format, allowing use of VIEW from any graphics terminal connected to the Network.

VIEW: A DISTRIBUTED SYSTEM FOR GRAPHICAL ANALYSIS OF LARGE DATA BASES*

Eric F. Harslem
Suzanne D. Landa

The Rand Corporation, Santa Monica, California

INTRODUCTION

Currently, numerous research efforts are involved in the investigation, simulation or analysis of natural phenomena. With the advent of supercomputers, numerical simulation of natural phenomena has become a feasible research tool. Unfortunately, not all such projects have the fiscal resources to acquire such a computing facility. Further, with the ability to run large-scale numerical simulations comes an equally large amount of simulated output which has to be comprehensively analyzed to achieve research goals.

This paper discusses a series of related solutions to the above problems. The first section deals with the acquisition of supercomputer facilities for research projects. The associated problem, storage of large volumes of data for analysis, is dealt with at the same time. The solutions in these two problem areas then become governing factors in the system design of the VIEW [1] graphical analysis system**, a flexible tool for the analysis of the large volumes of simulated data. The remainder of the paper describes the structure, properties and use of the VIEW system.

---

*This paper will be presented at the American Institute of Aeronautics and Astronautics Computer Network Conference, April 16-18, 1973.

**The development of VIEW was supported by the Department of Defense Advanced Research Projects Agency (ARPA).

As an example, we use the Climate Dynamics project sponsored by ARPA at the Rand Corporation.

There are several research projects in the ARPA, AEC, NASA communities to which these solutions are applicable.

RAND CLIMATE PROJECT

The central goal of the Rand climate project is to determine and quantify man's ability to accidentally or deliberately make significant modifications to the global climate. In pursuing this goal, the climate project makes use of several simulation models. The most important one, at present, is the Mintz-Arakawa Two-Level Atmospheric Circulation Model [2] (M/A model). The M/A model simulates the climate using a two-level 46 x 72 grid to represent the earth's atmosphere. The use of this model makes heavy demands for computing facilities -- both in terms of CPU processing and data storage.

In order to determine the probabilities of climate modification, a three month control run is made followed by several three-month experiments which include physical perturbations that might have been caused by man. The outputs of these M/A runs must then be compared for statistically significant differences [3]. The future needs of the project include year-long simulations with the possibility of increasing the fineness of the atmospheric grid.

At Rand, the main computer is an IBM 360/65. To simulate one day of real-time using the M/A model on the 360/65 requires five CPU hours using 400K bytes of high speed storage. Thus, a single three-month run would take 450 hours of CPU time -- at best a marginally realizable goal. Clearly, a series of year-long simulation would never be achieved.

Thus, the climate project encountered the first problem mentioned in the introduction -- it required supercomputer facilities for its work, but didn't have the financial resources to acquire them on a sole-user basis. Normal service bureaus were too limited in their revenue base to provide facilities of the magnitude required [4].

The solution to the acquisition of supercomputer resources was found in the nationwide ARPA Network.

ARPA NETWORK AND RESOURCE SHARING

PROPERTIES OF THE ARPA NETWORK

The ARPA Network* [5] is a nationwide network connecting 35 research institutions. The network is distributed in structure and heterogeneous in the types of computers it contains. Its present topology is shown in Figure 1.

The ARPA Network consists of two major parts -- the subnetwork and the host computers. The subnetwork consists of a set of 50-kilobit communication lines (heavy lines in Figure 1), and small message processors, called IMPs (squares in Figure 1), at each node. The communication lines provide for redundant paths between the nodes. The IMPs handle error checking, retransmission and traffic routing between the nodes removing all of the normal communications burden from the host computers. The host computers (ovals in Figure 1) include a wide variety of computer types and sizes ranging from the ILLIAC IV to PDP-11s and Terminal IMPs (TIPs).**

The main properties that attract users/servers to the ARPA Network are:

- o the isolation of communication problems in the subnetwork minimizes the effect on host operating systems
- o adaptively sharing high data rate communication lines yields fast transmission at low cost (approximately 30¢/million bits)

---

*The design of the ARPA Network is an outgrowth of many government-sponsored research projects in the 1960s [6]. Its topology is specified (and optimized) by Network Analysis Corporation while Bolt, Beranek and Newman is responsible for the actual software/hardware implementation.

**A TIP is a slightly enhanced IMP with a minimal host facility to provide basic terminal access to the Network.

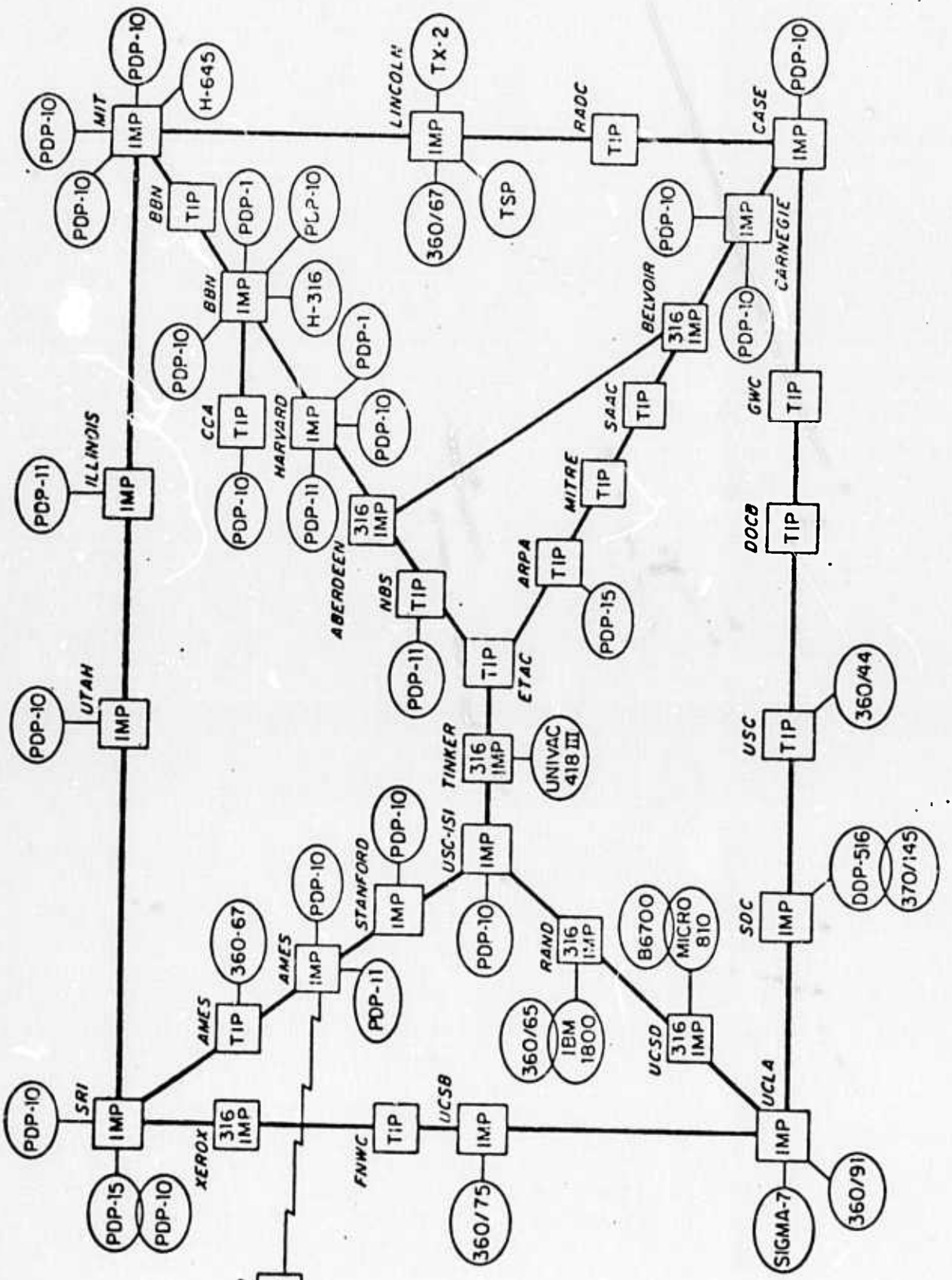
MIT
IMP
PDP-10
PDP-10
PDP-10
H-645
LINCOLN
IMP
TX-2
360/67
TSP
RADC
TIP
CASE
IMP
PDP-10
BBN
TIP
PDP-1
PLP-10
BBN
IMP
H-316
PDP-10
CCA
TIP
PDP-10
HARVARD
IMP
PDP-1
PDP-10
PDP-11
ILLINOIS
IMP
PDP-11
ABERDEEN
316 IMP
BELVOIR
316 IMP
PDP-10
IMP
PDP-10
CARNEGIE
SAAC
TIP
GWC
TIP
MITRE
TIP
NBS
TIP
PDP-11
ARPA
TIP
PDP-15
DOCB
TIP
UTAH
IMP
PDP-10
ETAC
TIP
TINKER
316 IMP
UNIVAC 418 III
USC
TIP
360/44
USC-ISI
IMP
PDP-10
PDP-10
STANFORD
IMP
PDP-10
AMES
IMP
PDP-11
AMES
TIP
360-67
RAND
316 IMP
PDP-10
B6700
MICRO 810
SDC
IMP
DDP-516
370/145
360/65
IBM 1800
UCSD
316 IMP
SRI
IMP
PDP-10
PDP-15
PDP-10
XEROX
316 IMP
FNWC
TIP
UCSB
IMP
360/75
UCLA
IMP
SIGMA-7
360/91
HAWAII
TIP

Figure 1.

- TIPS provide terminal access to the Network at a cost of about $30/month per terminal.

LARGE-SCALE RESOURCE SHARING

The properties of the ARPA Network described in the previous section have attracted a large nationwide user community. The fiscal base provided by this user community has made possible the dispersing of supercomputer facilities on a service center basis. For example, the UCLA 360/91 now derives more than 10% of its revenue from the ARPA Network user population.

The ability of the ARPA Network to share resources on a large scale has solved the problem of acquiring computer resources for research projects of all sizes. Many research projects now use the Network as their sole source computing facilities. In addition, the revenue base has provided facilities for on-line storage of extremely large data bases, an essential need in large-scale data analysis.

CLIMATE PROJECT USE OF THE ARPA NETWORK

Currently the climate project performs all of its large-scale computing on the UCLA IBM 360/91 via the ARPA Network. The computing power of the 360/91 provides a satisfactory host for the M/A model. On the 360/91, a day of real-time can be simulated in 1/2 hour of CPU time, making a three-month simulation reasonable. The output from a three-month run, however, is about 6×10^8 bits which are now stored on off-line disk packs. Since these packs must be mounted for data retrieval/analysis, obtaining the final results takes an undesirably long period of time. In addition, year long simulations are planned for the immediate future making the 360/91 only a temporary solution.

Over the next year, the heavy computing of the climate project will migrate to the NASA/AMES complex in northern

California. Since the AMES facility is also on the ARPA Network, the move will have a lessened effect on the researchers and programmers in the climate project.

Of principal importance at the AMES facility are the ILLIAC IV array processor [7] and the Unicon laser storage device [8]. Using the ILLIAC IV to run the M/A model will allow the simulation of a day in approximately 1-1/2 minutes of CPU time, making the goal of year long simulation feasible At the same time, the Unicon provides $.6 \times 10^{12}$ bits of on-line storage, removing the delays associated with accessing off-line data. Thus, the addition of the ILLIAC and Unicon will alleviate computer-related bottlenecks holding back the climate project.

The climate project now has the capability to run large simulations but is impeded by the magnitude of the data to be analyzed. This data analysis problem motivated the development of the VIEW system.

VIEW SYSTEM DESIGN

THE DATA PROBLEM

As the previous discussion has shown, the set of research projects involved in numerical simulation becomes increasingly bogged down in data analysis as the power of computers and complexity of simulation increases. One facet of the solution to this problem of data congestion is the ability to dynamically configure data for presentation at an on-line graphics terminal in an easy and natural fashion.

GENERAL GOALS OF THE VIEW SYSTEM

With the explosion of data to be analyzed, it is imperative to provide the researcher (the final decision-maker) the most direct access possible to the data and data analysis tools.* Therefore, the overriding consideration in the design of VIEW was that it be usable by a non-programmer with some minimal amount of instruction. To this end, we kept VIEW free of the normal "over-sophistication" that has relegated so many previous "ultimate" graphic systems to disuse.

At the same time, our contact with potential users indicated that the format and structure of displays would not fit into a rigid description. In fact, the exact requirements for a display would be arrived at when the data were being analyzed. Therefore, we had the added goal of being able to modify all aspects of a display dynamically at the time of display.

---

*In the case of the climate project, the old mode of operation was for a researcher to send a request to a programmer who would then do the data retrieval/analysis. In that mode of operation, a two-day turn around was considered exceptional.

In designing and implementing VIEW, we assumed that neither we nor the end users could exactly specify all the proper design criteria. Thus, in an attempt to avoid producing a useless system, we employed a modular structure with incremental implementation. As a result, various parts of VIEW were usable at an early stage, and in response to user suggestions, were enhanced with improvements that were carried over into subsequent stages of system design and implementation.

In summary, the major goals of VIEW were that it be simple to use and allow a flexible display format that could be specified on-line. Further implementation would proceed with corrective user feedback at the earliest possible points.

CONSTRAINTS ON THE VIEW SYSTEM DESIGN

In specifying the gross aspects of the VIEW system, several unusual constraints were involved. Since the general set of applications being addressed were dependent on the resources provided through the ARPA Network, the location of users and their data were liable to be both remote and variable. Thus, VIEW had to be able to

- be accessible by users at different locations in the Network
- be accessible from different types of terminals in the Network
- access data at different sites in the Network
- reside in an environment where long sessions and sporadic CPU usage were acceptable (i.e., not a batch environment).

STRUCTURE OF THE VIEW SYSTEM

To tailor the structure of VIEW to these constraints, it was divided into three major segments.

- o A graphics terminal I/O segment
- o A data retrieval segment
- o The main analysis program to handle user requests, instigate data retrieval and generate displays

Once VIEW was split into these segments, the constraints were easily satisfied by distributing the segments over hosts into the ARPA Network. Figure 2 shows a possible specific distribution of segments.

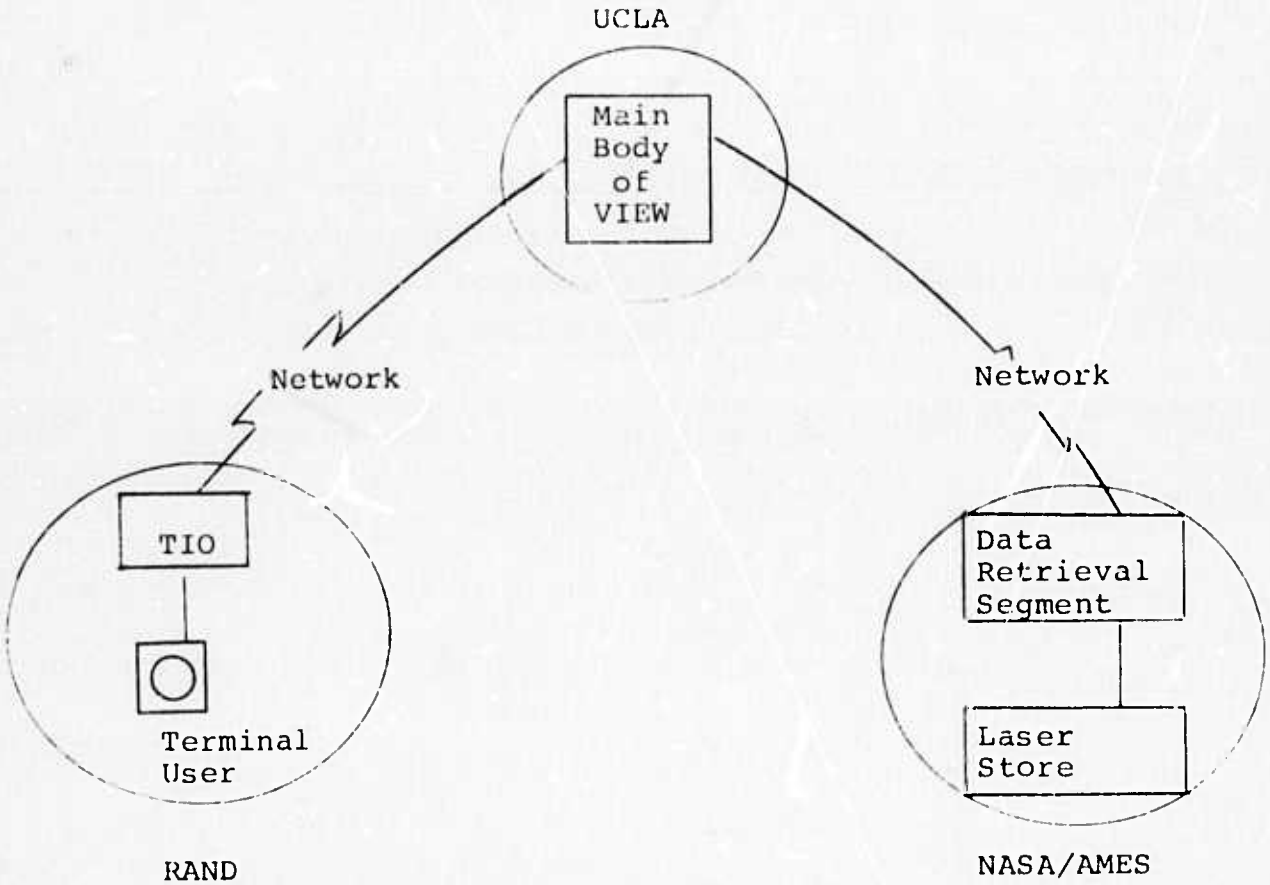

Figure 2.

Terminal Access

To allow easy use of VIEW from different terminals at different locations we took advantage of the program

sharing aspects of the ARPA Network. Normally, if a new user wanted to start using someone else's program, he would import it to his home system, modify it to run on his system and modify the I/O for his terminal. Under the Network program sharing philosophy, programs are designed with standard interfaces and accessed remotely while remaining in their natural computer environment. To this end, terminal input to VIEW is in the Network standard terminal input protocol [9] (TELNET) and display output to the terminal consists of order codes in Network Graphics Protocol Level 0 [10] (NGP0). (See Figure 3.)

Thus, when a new user/host wishes to use VIEW, he need have only these two protocol mechanisms available (as they are at most sites) and he can use the VIEW system without the usual program transfer/modification problems. This technique satisfies the constraints of easily allowing a variety of users access to VIEW.

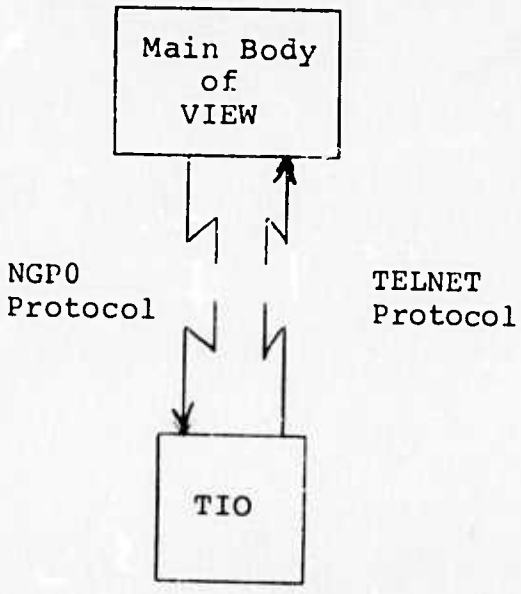

Figure 3.

Access to Data Bases

Satisfying the constraint of accessing various data bases proved to be a more difficult problem. Unlike the terminal I/O situation, there is no Network standard data retrieval protocol. Hence, the data retrieval interface in the main VIEW module was very explicitly defined so that each new remote data retrieval system could be handled by adding a new module.

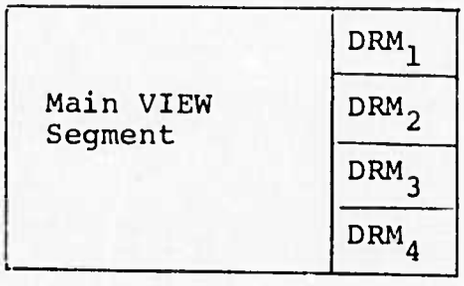

Figure 4.

As shown in Figure 4, there is a separate data retrieval module (DRM) for each different data retrieval system.

As part of the Rand climate project, a self-defining file and data retrieval system is being developed. The main use of this system will be data management on the 360/91 at UCLA and for the Laser Store at NASA/AMES. At present, interfacing with that system is the source of data for analysis by VIEW.

VIEW Operating System Environment

To allow protracted user sessions with fast response, it is necessary to run VIEW in a time-shared environment. It currently executes under the Time Sharing Option [11] (TSO) on the UCLA 360/91. However, since the main VIEW segment is written almost completely in FORTRAN, it could be transferred to another time-sharing system should the need ever arise. The main VIEW analysis program is further described in the following sections.

SUMMARY OF VIEW SYSTEM DESIGN

In summary, VIEW is a system with a simple but flexible user interface. Its implementation has been in increments to maximize the influence of user comments on the performance of the system. Segments of the system are distributed over hosts in the ARPA Network (although they could, in fact, all reside in the same host). Remote terminals of different types at different locations have access to VIEW via Network standard protocol. Remote data bases are accessed via the Network with specific VIEW modules to interface with various data retrieval systems.

VIEW ANALYSIS PROGRAM

The analysis program is the central part of the VIEW system; it acts on user inputs to retrieve data and present displays. The following sections discuss the analysis program from essentially a user's point of VIEW with discussions of the program's internals.

USER REQUIREMENTS

One user requirement was that displays be generated without specifying a long list of display parameters, while at the same time being able to modify all aspects of a display. To satisfy this requirement, VIEW deals with a set of variables that comprise a picture definition. The user can ignore these variables completely, in which case displays will be generated using default values for variables, or he can specify a subset of variables to modify the display to his own requirements.

In addition, once a display or display format was created users wanted to be able to save them for later use. Hence, a picture file facility is included in VIEW to allow storage and retrieval of picture definitions.

VARIABLES AND PICTURE DEFINITIONS

The collection of variables and data necessary to create and display a graph or other picture is called a picture definition. A picture definition can be stored in the user's picture file under a user-chosen name. The user's picture file includes a directory of the names and beginning record numbers of all the user's stored picture definitions, thus permitting direct access to a specified picture definition. The contents of the directory may be queried by the user. A stored picture definition consists of singly-linked blocks in a variable-length list structure. Such a structure is desirable since a picture may be described by as few as zero variables (all system defaults used) or as many as 20.

The picture definition consists of an arbitrary number of user-created variables. All these variables are transparent to the front-end processor of the analysis program. Certain of them will take on meaning in the context of the display processor invoked to create a picture. The remainder are annotation variables which the display process will assume to be supplemental text for display. In this fashion, the front-end is independent of display processes and need not be modified when display processes are added or changed.

Each variable has a set of attributes that can take on user-assigned values (see Table 1). When needed, display processes will provide default attributes for a variable. All attributes are identified by reserved words and some attributes may only take on reserved words as values. Not all attributes will have meaning for a given variable; however, all attributes exist with a default null value for each created variable. This organization greatly simplifies and generalizes the structural representation of a picture definition.

Table 1

ATTRIBUTES OF A VARIABLE

ATTRIBUTE	DESCRIPTION (Legal Name)
Name	A string specified by user or process
Value (Text)	String
Meaning	String
Type	SMALL, LARGE } for text
	SOLID, DASHED, TIC, SYMBOL } for figures
Value (Numeric)	4 Floating-point numbers
Orientation	VERTICAL HORIZONTAL

Attributes of a Variable

NAME:

Each variable has a character-string name and is created whenever that string appears on the left of an equal sign. The variable name may be reserved in the context of a display process. For example,

XLABEL = TEMPERATURE

creates a variable named XLABEL that is recognized by the display process GRAPH as a variable providing a label for the X-axis. As another example,

NOTE = HIGHEST PRESSURE POINT

creates an annotation variable named NOTE, which has no meaning to the GRAPH process and will be treated as supplemental text on the display.

VALUE(text):

The value of a variable is set to the string entered. For example,

TITLE VALUE = PRESSURE VS ALTITUDE

sets the TITLE to the string entered on the right hand side. The value attribute is assumed if no attribute is specified. Thus, the TITLE could also be specified by typing

TITLE = PRESSURE VS ALTITUDE.

MEANING:

The text explaining the meaning of a variable is set to the string entered. For example,

WINDOW MEANING = That portion of the physical display to be used.

The meaning attribute is a user aid so that functions of variables in the context of a display process can be interrogated on-line.

TYPE:

The type of the variable is set to the string entered. For example,

TITLE TYPE = LARGE

sets the character size of the title to large. The string entered for this attribute must be a reserved word from the specified set of alternatives or a symbol. Setting the TYPE attribute to a symbol is particularly useful for scatter diagrams. For example,

Y TYPE = +

specifies the plotting of the character '+' at each data point.

VALUE(numeric):

This value attribute serves different functions for different variables in the context of different display processes. For example, it can be referenced by

TITLE POSITION = 0,800

to move the location of the title of a display. Or it could be referenced as

XSCALE RANGE = 0,100,10

to specify the limits and increment for grid lines on the X-axis.

ORIENTATION:

The orientation of the variable is set to either HORIZONTAL or VERTICAL. For example, if

TITLE = TIME

TITLE ORIENTATION = VERTICAL

the title will be displayed as

T
I
M
E

VIEW USER LANGUAGE

The users involved in developing VIEW required that the input syntax be as simple as possible with a minimum of syntactic variations. To that end, the input to VIEW is limited to two syntactic forms.

(1) variablename [ø attributename] $\left\{\begin{matrix}=\text{string} \\ ?\end{matrix}\right\}$

(2) command [ø argumentlist]

Operations on Variables

All operations on variables and their attributes are performed using the first form with the exception of the DELETE and VARIABLES command. The set of possible operations on variables is shown in Table 2.

Operations on the Current Picture Definition and the Picture File

All of these operations are performed using the second syntactic type. They generally deal with storing and retrieving definitions or causing definitions to be used for display generation. These operations are explained in Table 3.

Table 2

OPERATIONS ON VARIABLES

Operation	Example	Consequence
Assign non-null value to attribute of a variable	A = 2	The character string "2" is assigned to the VALUE attribute of A. Note that A = 2 is identical to A VALUE = 2, i.e., if no attribute is specified, the VALUE attribute is assumed. If A does not currently exist, then a new variable, A, is created.
Assign null value	A =	The VALUE attribute of A is set to null.
Query an attribute of a variable	A ?	The value of the VALUE attribute of A is displayed.
	A TYPE ?	The current TYPE of A is displayed.
Purge a Variable	DELETE A	The variable A is purged from the picture definition.
Query all attributes of a variable	A ATTRIBUTES ?	The VALUE(text), MEANING, TYPE, VALUE(numeric), and ORIENTATION attributes for variable A are displayed.
List all variables in a picture definition	VARIABLES ?	The names of all variables in the current picture definition are displayed. Useful after retrieving a definition from a picture file.

Table 3

OPERATIONS ON PICTURE DEFINITIONS

Operation	Result
GET Picture definition	Retrieves a copy of the named picture definition, which becomes the current picture definition. All subsequent operations on variables affect this copy of the picture definition.
PUT Picture definition	Saves current picture definition under the specified name.
PURGE Picture definition	Delete a picture definition from the picture file.
RENAME Old Picture definition, new Picture definition	The old picture definition name is replaced by the new picture definition name.
DIRECTORY	A list of the user's picture definitions is displayed.
INITIALIZE	Deletes all variables from the current picture definition.
GRAPH	Invokes a program which uses the current picture definition to generate an X-Y graph on the user's terminal.
CONTOUR	Generates a contour map using the current picture definition.
CHART	Generates a bar chart on the user's terminal.
PLOT	Generates a vector plot.

ORGANIZATION OF THE VIEW ANALYSIS PROGRAM

VIEW was designed to allow incremental implementation thus facilitating debugging and modification. The modules include a front-end processor, display generation programs and data retrieval modules. The relationships of these parts and their subparts are shown in Figure 5.

The Front-End Module

The command decoder accepts and parses the user's input to determine the appropriate module to invoke. The accessor reads picture definitions from the disk picture file into core, stores the current picture definition on disk, renames and purges picture definitions. The accessor may be queried for a list of the user's picture definitions. The picture definition editor is responsible for editing variables in the picture definition according to user requests. For example, this module would handle requests to change the value of a variable. The executor determines from the user's command which display process to invoke.

Display Generation Processes

Each of these processes initiates the retrieval process and generates a different type of display. Types of displays to date are graphs, contours, vector plots, and bar charts. These programs use the retrieved data, the user-specified variables in the current picture definition and, when necessary, the built-in defaults to display the requested picture.

Retrieval Modules

The retrieval modules use the retrieval variables in the current picture definition to selectively retrieve data from the user-specified data base. If retrieval variables do not exist, the retrieval module defaults to using all the data for display.

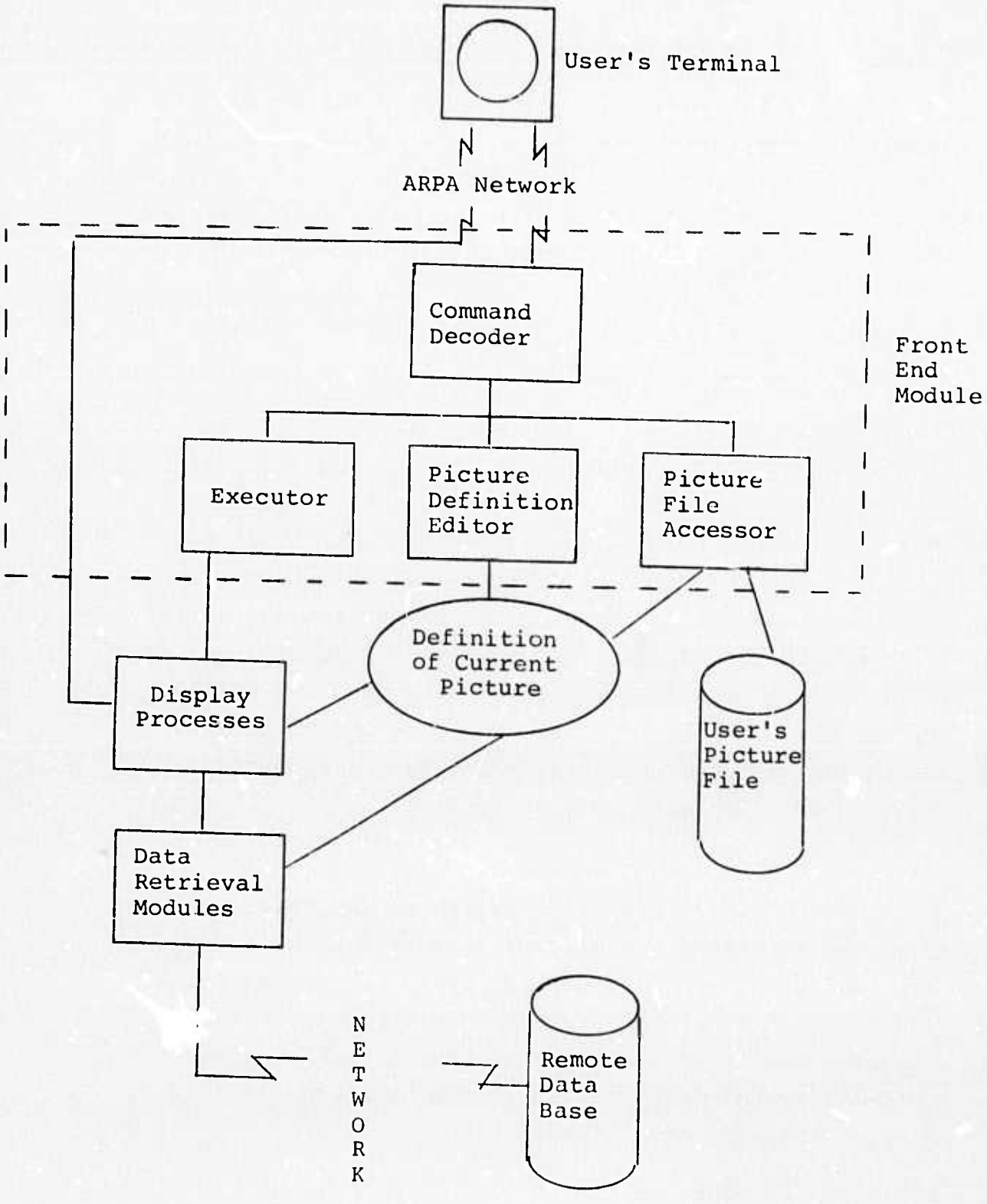

Figure 5. Elements of the VIEW Analysis Program

SCENARIO FOR VIEW USE

To use VIEW from a graphic console at Rand the user enters the following sequence of commands.

1)	NETWORK	Connect to the Rand Network Access Program (NAP) (provides TELNET functions).
2)	SET GRAPHICS	Enables NGP0 interpreter for graphic output to the terminal.
3)	TELNET UCLA-CCN	Makes a duplex connection to UCLA TSO. NAP automatically handles TSO logon.
4)	EXEC VIEW	Start program which asks for next two inputs.
5)	picture file name	Specifies file for picture definition storage/retrieval.
6)	data source	Specifies source of data for analysis.

At this point the user may request the VIEW functions previously described. The only information a user need supply for a display is the type he desires. To display a graph of his data he need only enter

GRAPH.

Since no data qualifiers have been provided, this command directs the data retrieval package to examine the data definition block and to read in up to 2 dimensions of data. If the data have only one dimension the GRAPH module will plot it against its index. Also, since the data descriptor provides the name of each variable the GRAPH process can generate default axis labels and a display title. The data extremes are used to label the axis end points. A graph such as the one shown in Figure 6 will appear on the screen.

At this point, the user has a minimal X-Y graph as generated by the default GRAPH variables. He can now

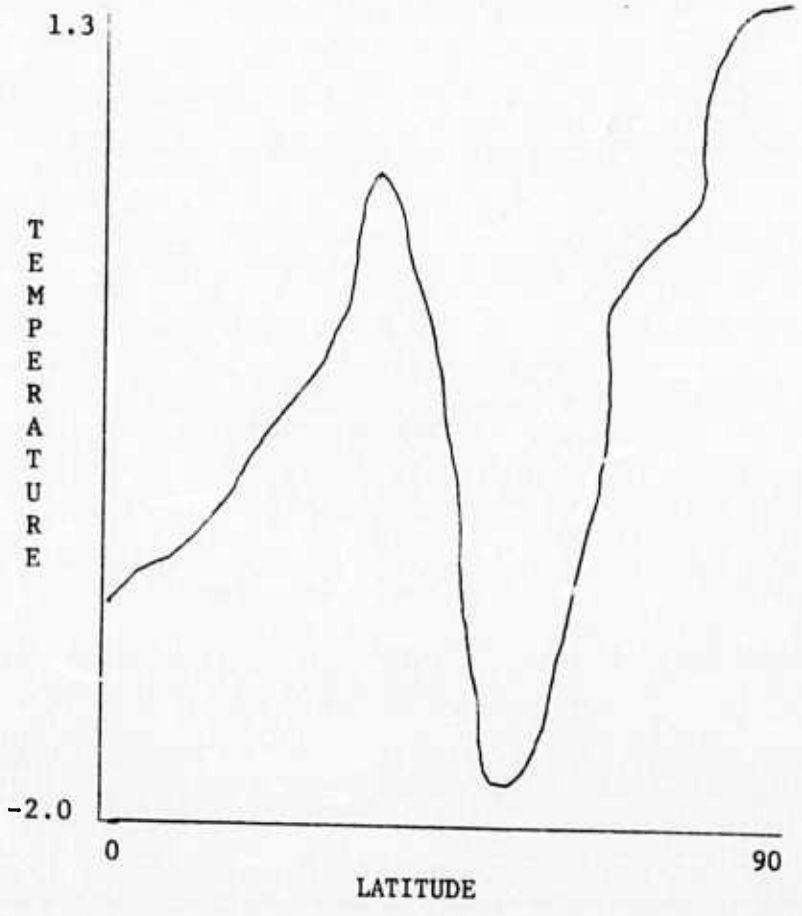

Figure 6

enhance the display by specifying his own values for some of these variables.

To examine a specific part of the curve in more detail with grid lines for registration he can enter

```
YSCALE RANGE = 0,1.0,.25
XSCALE RANGE = 30,50,5
```

and to add more descriptive legends and remarks he could enter

```
TITLE = ZONAL AVERAGE OF UPPER LEVEL, DAYS 31-60
YLABEL = TEMPERATURE CHANGE (°C)
NOTE = BLACK CLOUD MINUS CONTROL
NOTE POSITION = 100,850
```

By then entering "GRAPH" he would receive the display shown in Figure 7.

ZONAL AVERAGE OF UPPER LEVEL, DAYS 31-60
BLACK CLOUD MINUS CONTROL

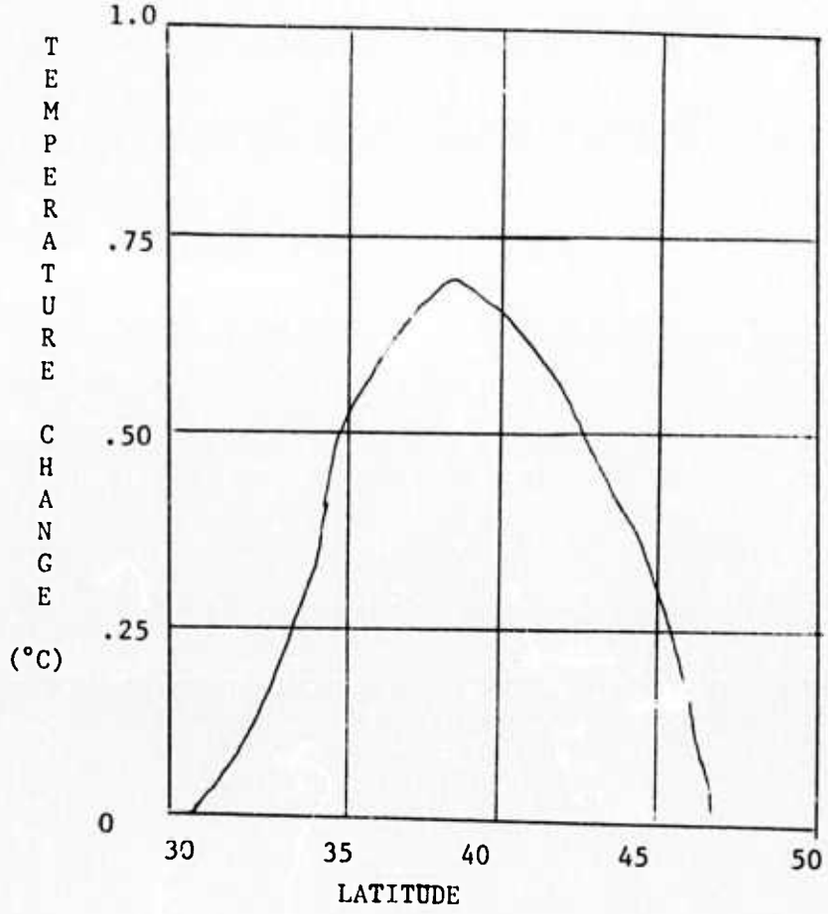

Figure 7

DISCUSSION

VIEW is now in its initial stages of real productive use by the climate project at Rand. The users feel that it generally satisfies their requirements; however, several rounds of improvements and modification have been made in response to their criticism.

The main problem in the use of VIEW at present is the fact that the data base still resides on off-line disk packs at UCLA. Thus, a user has a two-step process to get actual displays -- retrieving a subset of data to on-line storage, then displaying it with VIEW. The move to the Laser Store and the self-defining data retrieval system should eliminate this problem.

In summary, the initial use of VIEW has been encouraging; however, final analysis of its effectiveness in solving the analysis problems involved must wait until smoother access to data can be provided.

REFERENCES

1. Harslem, E. F., and J. F. Heafner, VIEW: Status Report, The Rand Corporation, R-1069-ARPA, August 1972.

2. Gates, W. L., et al., A Documentation of the Mintz-Arakawa Two-Level Atmospheric General Circulation Model, The Rand Corporation, R-877-ARPA, December 1971.

3. Warshaw, M., The Sensitivity and Comparison of General Atmospheric Circulation Models, The Rand Corporation, P-4933, November 1972.

4. Harslem, E. F., and Heafner, J. F., Aspects of Large Scale Resource Sharing Through Networks of Computers, The Rand Corporation, P-4833, May 1972.

5. Roberts, L. G., "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, Vol. 36, 1970, pp. 543-550.

6. Baran, Paul, On Distributed Communications, The Rand Corporation, RM-3420-PR, August 1964.

7. Barnes, G. H., et al., "The ILLIAC IV Computer," IEEE Transactions on Computers, Vol. C-17, No. 8, August 1968, pp. 746-757.

8. McFarland, K. and M. Hashiguchi, "Laser Recording Unit for High Density Permanent Data Storage," AFIPS Conference Proceedings, Vol. 33, Part 2, 1968, pp. 1369-1380.

9. TELNET Protocol, Stanford Research Institute, NIC #9348, April 1972.

10. Graphics Protocol - Level 0, Stanford Research Institute, NIC #8302, January 1972.

11. Time Sharing Option - Command Language Reference, IBM Corporation, Poughkeepsie, N. Y., SRL #GC28-6732, September 1971.